REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	31 October 2002	Final Report 08/01/01 – 12/31/01

4. TITLE AND SUBTITLE
Distributed Algorithms for Sensor Fusion

5. FUNDING NUMBERS
G DAAD19-01-1-0724

6. AUTHOR(S)
David Meyer
Jeffrey Remmel

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Regents of the Univ of Calif,
Univ of Calif, San Diego
9500 Gilman Drive, 0112
La Jolla, CA 92093-0112

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

42724.2-MA

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The goal of this project was to develop algorithms which will process the information collected by networks of large numbers of low power, low bandwidth sensors. We have analyzed failure modes for simple cases of hierarchical networks, developed foundational results in logic programming and hybrid control, and constructed a simulation package. We would like to continue with this project which, despite a slow start, is now - after 3 quarters - showing results.

14. SUBJECT TERMS
sensor fusion

15. NUMBER OF PAGES
11

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	**UNCLASSIFIED**	**UNCLASSIFIED**	**UL**

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

1. Overview

Recent developments in sensor device technology offer the possibility of deploying hundreds of thousands or even millions of sensors and linking them wirelessly. One can envision networks at two extremes [1]:

1. Low cost, low power, low bandwidth, battery powered microsensors, with limited processing and communication capabilities, networked locally (with respect to geography).

2. Sensors, processors, and actuators embedded in sensing and weapons platforms, vehicles or soldiers, with far greater processing and communications capabilities, organized in dynamical networks.

Each type of network will generate immense quantities of data which must be coordinated, interpreted and acted upon. While optimal data fusion algorithms have been developed for small, typically unconstrained, networks of sensors [2–4], very little is known about data fusion in networks of 10^5–10^6 sensors. The goal of this project is to develop general principles and specific algorithms for (near) optimal solutions to this problem. To build a complete system, of course, our results would have to be integrated with smaller scale algorithms on the single or few device level (for, *e.g.*, target recognition and classification).

We have concentrated on *localized* processing algorithms for several reasons. First, in the low power, low bandwidth network listed above, processing is *constrained* to be local. Second, even with the greater capabilities in the embedded sensor network, it is computationally infeasible (NP-complete) to fuse all the available data optimally [5]. Nevertheless, analogies with communication theory [6,7], specifically the FFT and decoding large block length codes, suggest that efficient and near optimal solutions can be achieved with local, hierarchical algorithms. Third, in dynamical networks, and particularly when coupled to actuators, robustness to network damage and responsiveness to local conditions suggest advantages to localized processing.

2. Accomplishments

2.1. Original goals

2.1.A. Majority algorithms for sensor fields

We began by considering a relatively simple scenario: a field of sensors randomly distributed across some area, tasked to report the presence of a tank battalion, for example, but to ignore a single or small number of tanks. Suppose that the sensors have been distributed randomly, from the air, perhaps, as shown in green in Figure 1. In the low power, low bandwidth extreme, the sensors can detect the presence of a tank by, *e.g.*, its acoustical signature [8,9], and can broadcast that detection locally. Suppose further that there are a smaller number of aggregating devices, shown in blue, with greater processing and communications capabilities (possibly including GPS locators [10]) which receive the

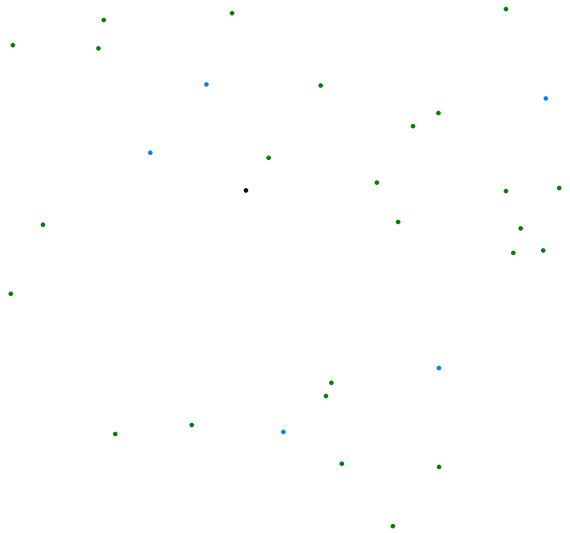

Figure 1. A random distribution of 25 sensors (green), 5 aggregating devices (blue) and 1 root device (black).

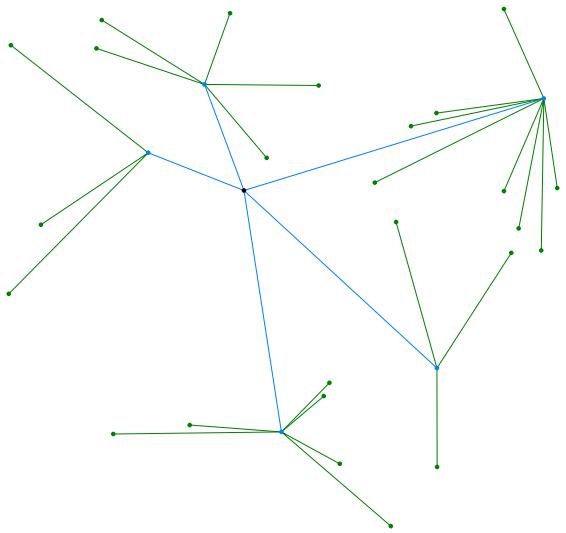

Figure 2. Each sensor is connected to the nearest aggregating device, each of which connects to the root.

signals from the nearest sensors and broadcast the results to a third layer of the hierarchy, which aggregates their reports. Figure 2 shows the resulting communication network for the random distribution of Figure 1; in this example there are 25 sensors, connected by green lines to 5 aggregators, connected by blue lines to 1 ('root') device at the third level of the hierarchy.

For devices with specific operating characteristics, one would want to optimize the network architecture and processing, given costs associated with latency, and with false positive and false negative responses. Our original goal was first to develop general principles which would apply to a variety of possible devices and costs—which could change over time, and in response to strategic considerations, respectively. Some general principles can be deduced from simple models: The simplest decision rule the network of Figure 2 might attempt to implement is MAJORITY [11]. That is, if a majority of the 25 sensors detect a tank, the report of the sensor field should be BATTALION, otherwise it should be NO BATTALION. (We assumed for our initial simulations that the individual sensors always give correct reports.) Notice that in this scenario we are asking the sensor field to provide a crude solution to the 'disaggregation' problem. That is, under the assumption that the sensors are distributed at a spatial scale such that each responds to $O(k)$ tanks on the average (for $k < 1$), it is approximately distinguishing between numbers of tanks greater and less than $25/2k$. It is doing so, furthermore, with a *network* algorithm—without the signal processing more commonly applied at the single or few device level to disaggregate—since we are working in the limited computation, low bandwidth scenario.

Figure 3 illustrates an event to which the network responds correctly: 14 sensors—shown in red—fire, which causes 3 of the aggregators to fire; since this is more than half,

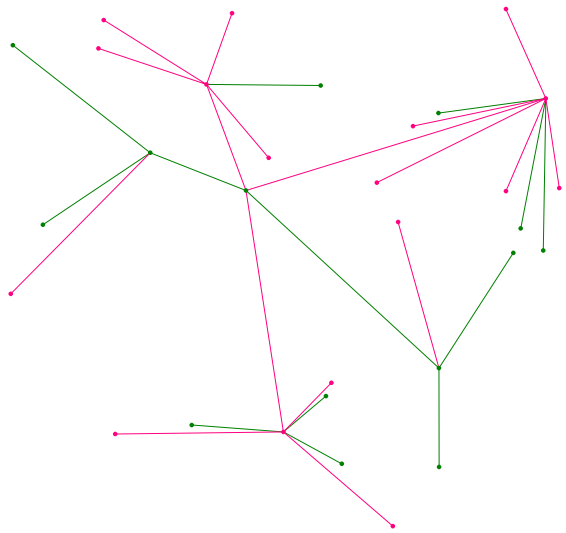

Figure 3. 14 of the sensors—shown in red—fire and the network correctly reports the presence of a battalion.

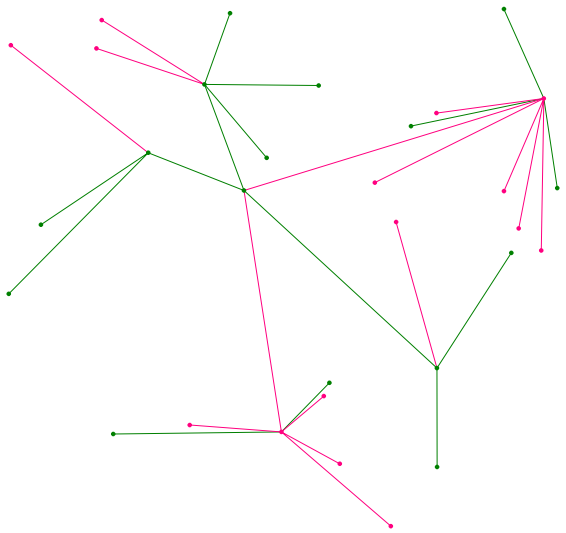

Figure 4. 13 of the sensors—shown in red—fire but the network incorrectly reports that no battalion is present.

the root correctly reports BATTALION. But this network is not infallible, even when we assume that all the devices function correctly. Figure 4 illustrates an event to which the network responds incorrectly: 13 sensors—shown in red—fire, but they only cause 2 of the aggregators to fire, so the root responds NO BATTALION, a false negative.

In our preliminary investigations, instances like this were identified by simulation. We have shown subsequently that the number of these kinds of errors can be calculated explicitly. The results are shown in Figure 5, where the probability of an error is plotted as a function of the number of sensors which fire for the sensor network of Figures 2–4. For small or large numbers of firing sensors the network MAJORITY algorithm always correctly identifies BATTALION or NO BATTALION, but for numbers near the $25/2k$ threshold the error rate climbs to more than 1/3. These results are detailed in [12].

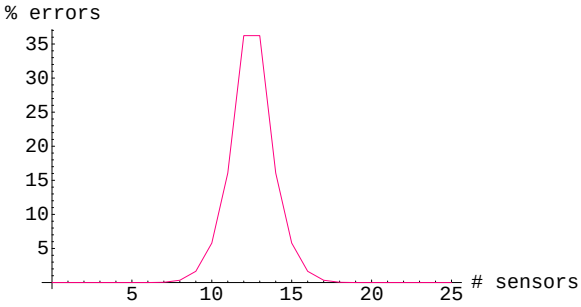

Figure 5. The percentage of erroneously processed events, plotted as a function of the number of sensors which fire, for the sensor network shown in Figures 2–4.

We have begun to produce a theoretical framework for this kind of analysis by identifying an equivalence with certain kinds of error correcting codes. Notice that each aggregating device can be thought of as performing error correction on a repetition code: if there are 5 sensors reporting to an aggregating device, for example, the corresponding repetition code encodes a bit $b \in \{0, 1\}$ as $bbbbb$. In the coding context, errors may flip

some of the bits during transmission, and then MAJORITY decoding correctly identifies b as long as the number of corrupted bits is no more than $2 = (5-1)/2$, where 5 is the Hamming distance between the codewords `00000` and `11111`. In [13] we show that a hierarchical network algorithm can be interpreted as decoding a *concatenated* error correcting code. The largest number of firing sensors which is never misidentified as BATTALION (6 in Figure 5) is $(d-1)/2$, where d is the *distance* of the concatenated code.

2.1.B. Extending logic programming for distributed sensor algorithms

We have also been working on foundational issues supporting the development of general algorithms for sensor fusion. A first step is to extend the logic programming paradigm so that it can be used as a platform for high level reasoning agents for sensor fusion problems. Papers [14–16] are concerned with various questions motivated by recent developments in Knowledge Representation, especially the appearance of a new generation of systems [17–19] based on the so-called Answer Set Programming (ASP) paradigm [20–22]. Answer Set Programming is a recent attempt to develop a logic-based formalism that can be used in a variety of high level reasoning tasks. One theme of this work is develop efficient programming engines that will allow the user to solve a variety of NP-search problems [23] of the type that arise in variety of pattern matching problems in the context of sensor fusion. For example, if we have a partial view of military vehicle, how can we match the salient features that we can observe to known vehicles to decide exactly the type of vehicle with which we are dealing?

More formally, a search problem is a set $\mathcal{S}$ of finite instances such that, given any instance $I \in \mathcal{S}$, there is a set S_I of solutions to $\mathcal{S}$ for instance I. It is allowed that S_I is the empty set. For example, the search problem may be to find Hamiltonian paths in a graph. Thus, the set of instances of the problem is the set of all finite graphs. Then, given any instance, *i.e.*, a graph G, S_G is the set of all Hamiltonian paths of G. We say that an algorithm solves the search problem $\mathcal{S}$ if it returns a solution $s \in S_I$ whenever S_I is non-empty and it returns the string "empty" otherwise. We say that a search problem *S is in* NP if there is such an algorithm which can be computed by a non-deterministic polynomial time Turing machine. We say that search problem *S is solved by a uniform logic program* if there exists a single logic program $P_{\mathcal{S}}$, a polynomial time extensional data base transformation function $edb_{\mathcal{S}}$ and a polynomial time solution decoding function $sol_{\mathcal{S}}(\cdot,\cdot)$ such that for every instance I in $\mathcal{S}$,

1. $edb_{\mathcal{S}}(I)$ is a finite set of facts, *i.e.*, clauses with empty bodies and no variables,

2. whenever $sol_{\mathcal{S}}(I)$ is non-empty, $sol_{\mathcal{S}}(I,\cdot)$ maps the set of stable models of the $edb_{\mathcal{S}}(I) \cup P$ onto the set of solutions S_I of I, and

3. if $sol_{\mathcal{S}}(I)$ is empty, then $edb_{\mathcal{S}}(I) \cup P$ has no stable models.

Our work has focused mostly on one particular ASP formalism, specifically, the Stable Semantics for Logic Programs (SLP) [24], and its extensions. The underlying methods of ASP are similar to those used in Logic Programming [25] and Constraint Programming

[26,27]. That is, like Logic Programming, ASP is a declarative formalism and the semantics of all ASP systems are based on logic. Like Constraint Programming, certain clauses of an ASP program act as *constraints*. There is a fundamental difference between ASP programs and Constraint Logic programs, however. That is, in Constraint Programming, the constraints act on individual elements of the Herbrand base of the program while the constraint clauses in ASP programs act more globally in that they place restrictions on what subsets of the Herbrand base can be acceptable answers for program. For example, suppose that we have a problem Π whose solutions are *subsets* of some Herbrand base H. In order to solve the problem, an ASP programmer essentially writes a logic program P that describes the constraints on the subsets of H which can be answers to Π. The basic idea is that the program P should have the property that there is an easy decoding of solutions of Π from stable models of P and that all solutions of Π can be obtained from stable models of P through this decoding. The program P is then submitted to an ASP engine such as *smodels* [18], `dlv` [19] or DeReS [17] which computes the stable models of the program P. Currently, the systems based on ASP paradigm are being tested on the problems related to planning, product configuration, combinatorial optimization problems and other domains.

Our approach in this project is to use ASP as a platform for pulling together the information contributed from a variety of sensors in a network monitoring a region for possible enemy activity and allowing the system to "search" for fundamental patterns to help identify individual objects, patterns of movement of groups of objects, and patterns that suggest danger and the need for immediate actions. Part of our goal is to extend this formalism to allow the system to reason about geographical regions or restricted areas of space which we call Spatial Logic Programming [15]. We hope to have a student working on implementing such a system on top of existing ASP engines this summer.

2.1.C. Hybrid system optimization techniques for agent-based sensor algorithms

We have also continued our development of hybrid systems and, in particular, the development of a distributed agent-based optimization network called the Multiple Agent Hybrid Control Architecture (MAHCA), first developed by Kohn and Nerode [28]. This work is described in [29,30]. The general framework in which MAHCA operates is shown in Figure 6. In a typical application that we have in mind, the distributed process consists of a global network of sensors and each agent monitors a small local network of sensors.

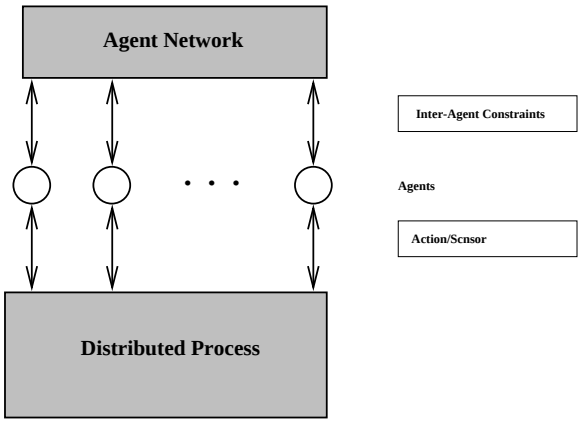

Figure 6. MACHA framework.

The Multiple Agent Hybrid Control Architecture is implemented as a distributed system composed of agents, and a communication network called the logic communication

network. The architecture realizing this system operates as an on-line distributed theorem prover. In general, the MAHCA architecture interacts with the system it monitors at a series of update times $\Delta_1 < \Delta_2 < \ldots$. These update times are a function of the application and the next update time is determined by the interaction between the system and the agents. At an update time, each active agent will receive information from a certain suite of sensors and generate estimation or control actions as a side effect of proving an existentially quantified subtheorem (lemma) which encodes the model of the system as viewed by the agent. The conjunction of lemmas at each instant of time encodes the desired behavior of the entire network. The number of agents in the network is variable. That is, the system can spawn new agents and deactivate agents as a function of system demand. Each agent of MACHA consists of five modules:

Planner: Constructs and repairs the agent's optimization criterion.

Inferencer: Determines whether there is a nonempty solution set for the agent's optimization problem. If there is such a solution set, the Planner infers the appropriate control actions, new state information and inter-agent information.

Adapter: Repairs failure terms and computes correction terms.

Knowledge Base: Stores and updates the agent's knowledge.

Knowledge Decoder: Receives and translates data from the other agents.

In general, a hybrid system has a hybrid state, the simultaneous dynamical state of the system and all digital control devices. Properly construed, the hybrid states will form a differentiable manifold which Kohn and Nerode call the *carrier manifold* of the system. To incorporate the digital states as certain coordinates of points of the carrier manifold, we "continualize" the digital states [30,31]. That is, we view the digital states as finite, real-valued, piecewise-constant functions of continuous time and then take smooth approximations to them. This allows us to consider logical and differential or variational constraints on the same footing, each restricting the points allowed on the carrier manifold. This also allows us to use classical control and differential geometric techniques to develop algorithms for sensor-agent networks. Part of our effort in the next two years will be devoted to adapting such methods to sensor fusion problems like those considered by Kohn, Nerode and Remmel [32].

Finally, *index sets* are ways to measure the complexity of many problems in logic, combinatorics, and computer science. In [33] we investigate the connections between index sets for ω-languages and index sets for computable analysis.

2.2. Revised goals

In January 2002 we met with John Lavery, who encouraged us to consider some modified versions of the problems described in §2.1. In particular, he emphasized the disaggregation problem, in the context of a substantially denser array of sensors than envisioned in §2.1.A

($\gg 1$ sensor/TANK rather than < 1 sensor/TANK) but with positive failure probabilities. The simplest scenario consists of a sensor field with a single aggregating device (*i.e.*, the network consists of exactly those sensors within transmission range of an aggregating device) with the sensors dense compared to the detection area for a TANK. If we assume, furthermore, that each sensor reports correctly with probability $p < 1$, independently, this becomes a Bayesian estimation problem. That is, suppose k sensors fire. We wish to determine $\Pr(\text{TANK} \mid k)$. Simple algebra gives:

$$\begin{aligned}\Pr(\text{TANK} \mid k) &= \frac{\Pr(k \mid \text{TANK}) \Pr(\text{TANK})}{\Pr(k)} \\ &= \frac{\Pr(k \mid \text{TANK}) \Pr(\text{TANK})}{\Pr(k \mid \text{TANK}) \Pr(\text{TANK}) + \Pr(k \mid \text{NO TANK}) \Pr(\text{NO TANK})}.\end{aligned}$$

For this simple scenario, we can compute $\Pr(k \mid \text{TANK})$ and $\Pr(k \mid \text{NO TANK})$ analytically. $\Pr(\text{TANK})$, however, must be an external input to the algorithm, specified perhaps by the current threat conditions or some other intelligence. Given this input and the number k of sensors which fire, we can compute the probability that there is a TANK present in the sensor field.

In general, if there is a set of events E_i among which we wish to distinguish, and possible sensor network outputs S_j, the Bayes formula is

$$\Pr(E_i \mid S_j) = \frac{\Pr(S_j \mid E_i) \Pr(E_i)}{\sum_i \Pr(S_j \mid E_i) \Pr(E_i)}.$$

For more complicated networks, *i.e.*, with more than one aggregating device, or more levels, the conditional probabilites for various sensor network outputs given various events may not be possible to calculate analytically. With this in mind we have begun to construct a simulator which can accommodate large numbers of sensors and aggregating devices, arranged randomly or not, networked into a multilevel hierarchy. The simulator incorporates a flexible error model for the individual sensors, and allows simulation of various network algorithms. Figure 7 shows a screenshot of the simulator. So far we have applied it only to the simple scenario described above, in order that we could verify the code by comparison with the analytic solutions. The next step is to simulate more complicated scenarios. In particular, we believe that we can address the disaggregation problem with this system.

3. Future directions

This project has been running less than a year (it began in August 2001), and despite a slow start, we believe that we have by now made substantial progress. We are eager to continue with various aspects of this project: not only the immediate next steps mentioned in §2.1.B and §2.2, but also several new directions. David Meyer, Jeff Remmel and Victor Marek have been meeting this spring to discuss integrating the three components of this project: the explicit network algorithms described in §2.1.A and §2.2, the logic programming formalism described in §2.1.B, and the hybrid control architecture described in §2.1.C. We have also

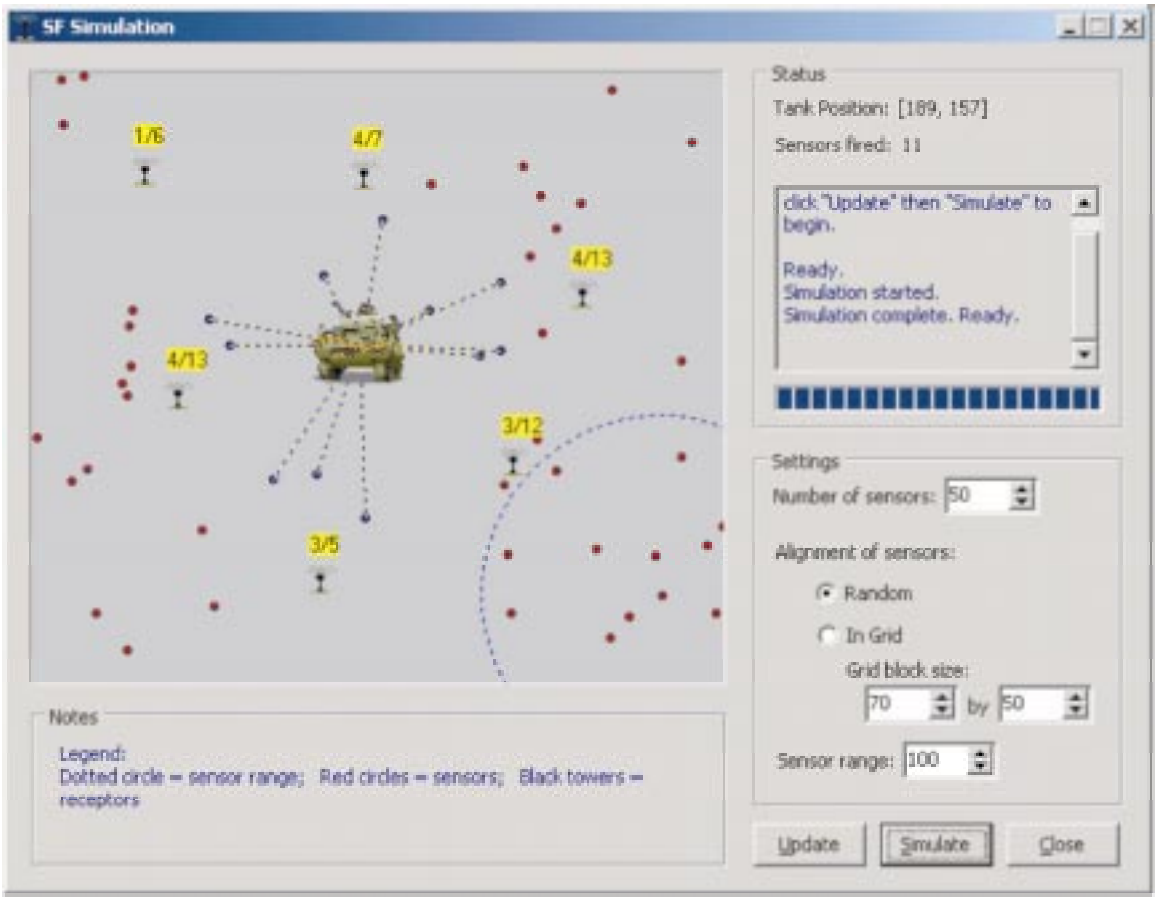

Figure 7. Screenshot of the sensor network simulator.

met recently with two defense industry scientists: Harry Schmitt and Ross Rosenwald of Raytheon's Missile Systems division. They are also interested in collaborating with us to develop distributed sensor algorithms.

4. Publications and references

[1] United States Army Training and Doctrine Command, *Knowledge and Speed: Battle Force and the U. S. Army of 2025, 1998 Annual Report on The Army After Next Project to the Chief of Staff of the Army* (Ft. Monroe, VA: TRADOC 1998).

[2] E. Waltz and J. Llinas, *Multisensor Data Fusion* (Boston: Artech House 1990).

[3] I. R. Goodman, R. P. S. Mahler and H. T. Nguyen, *Mathematics of Data Fusion* (Boston: Kluwer 1997).

[4] P. K. Varshney, *Distributed Detection and Data Fusion* (NY: Springer 1997).

[5] J. Tsitsiklis and M. Athans, "On the complexity of decentralized decision making and detection problems", *IEEE Trans. Automatic Control*, **30** (1985) 440–446.

[6] D. Middleton, *An Introduction to Statistical Communication Theory* (New York: McGraw-Hill 1960).

[7] T. L. Gabrielle, "Information criterion for threshold determination", *IEEE Trans. Inform. Theory*, **6** (1966) 484–486.

[8] N. Srour and J. Robertson, "Remote netted acoustic detection system", Army Research Laboratory Technical Report ARL-TR-706 (1995).

[9] K. Eom, M. Wellman, N. Srour, D. Hillis and R. Chellappa, "Acoustic target classification using multiscale methods", preprint (1997).

[10] P. Janiczek, *Global Positioning System* (Washington, DC: The Institute of Navigation 1980).

[11] A. Ziv and J. Bruck, "Checkpointing in parallel and distributed systems", in A. Y. Zomaya, ed., *Parallel and Distributed Computing Handbook* (New York: McGraw-Hill 1996) 274–302.

[12] D. A. Meyer, "Failure rates in distributed sensor algorithms", in preparation.

[13] D. A. Meyer, "A code theoretic approach to distributed sensor algorithms", in preparation.

*[14] V. W. Marek and J. B. Remmel, "On the expressibility of stable logic programming", submitted to *Theory and Practice of Logic Programming*.

*[15] H. A. Blair, V. W. Marek and J. B. Remmel, "Spatial logic programming", *Proceeding of the 2001 World Multiconference on Systemics, Cybernetics and Informatics*.

*[16] V. W. Marek and J. B. Remmel, "On logic programs with cardinality constraints", in S. Benferhat and E. Giunchiglia, eds., *Proceedings of 9th International Workshop on Nonmonotonic Reasoning*, 219–228.

[17] P. Cholewiński, V. W. Marek and M. Truszczyński, "Default reasoning system DeReS", in *Proceedings of KR-96* (Morgan Kaufmann 1996) 518–528.

[18] I. Niemelä and P. Simons, "Efficient implementation of the well-founded and stable model semantics", in *Proceedings of JICSLP-96* (Cambridge: MIT Press 1996).

[19] T. Eiter, N. Leone, C. Mateis, G. Pfeifer and F. Scarcello, "A deductive system for nonmonotonic reasoning", in *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer Lecture Notes in Computer Science **1265** (1997) 363–374.

[20] I. Niemelä. "Logic programs with stable model semantics as a constraint programming paradigm", in *Proceedings of the Workshop on Computational Aspects of Nonmono-*

tonic Reasoning (1998) 72–79.

[21] V. W. Marek and M. Truszczyński, "Stable models and an alternative logic programming paradigm", in *The Logic Programming Paradigm*, Series Artificial Intelligence (Berlin: Springer-Verlag 1999) 375–398.

[22] V. Lifschitz, "Action languages, answer sets and planning", in *The Logic Programming Paradigm*, Series Artificial Intelligence (Berlin: Springer-Verlag 1999) 357–373.

[23] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness* (W. H. Freeman 1979).

[24] M. Gelfond and V. Lifschitz, "The stable semantics for logic programs", in *Proceedings of the 5th International Symposium on Logic Programming* (Cambridge, MA: MIT Press 1988) 1070–1080.

[25] K. R. Apt, "Logic programming", in *Handbook of Theoretical Computer Science* (Elsevier 1990) 475–574.

[26] J. Jaffar and M. J. Maher, "Constraint logic programming: a survey", *J. Logic Programming* **19** (1994) 503–581.

[27] K. Marriott and P. J. Stuckey, *Programming with Constraints: An Introduction* (Cambridge, MA: MIT Press 1998).

[28] W. Kohn and A. Nerode, "Multiple Agent Hybrid Control Architecture", in R. L. Grossman, A. Nerode, A. Ravn and H. Rischel, eds., *Hybrid Systems*, Lecture Notes in Computer Science **736** (Berlin: Springer-Verlag 1993) 297–315.

*[29] W. Kohn and J. B. Remmel, "Multi-echelon inventory planning system I: a muliple agent hybrid control implementation", *Proceeding of 2001 World Multiconference on Systemics, Cybernetics and Informatics*.

*[30] W. Kohn and J. B. Remmel, "Multi-echelon inventory planning system II: contiualization", *Proceeding of 2001 World Multiconference on Systemics, Cybernetics and Informatics*.

*[31] H. Blair and J. B. Remmel, "Hybrid automata: convergence spaces and continuity", *Proceeding of 2001 World Multiconference on Systemics, Cybernetics and Informatics*.

[32] W. Kohn and J. B. Remmel, "Implementing sensor fusion using a cost based approach", *Proceedings of the 1997 American Conference of Control*.

*[33] D. Cenzer and J. B. Remmel, "Index sets for ω-languages", to appear in *Math. Logic Quarterly*.

The *s indicate manuscripts reporting work performed or published as part of DAAD19-

01-1-0724. Three copies of each are included.

5. Personnel

This grant has partially supported the work of David Meyer and Jeff Remmel. Louka Dlagnekov has been working with David Meyer since March 2002 on an undergraduate research project involving designing, writing and applying a computer simulation of distributed sensor algorithms. Jeff Remmel has been working with Douglas Cenzer, and with Victor Marek and Wolf Kohn, who have been involved in discussions on applying logic programming and hybrid control to algorithms for distributed sensors and actuators.